JP2002091780 A
SYSTEM AND METHOD FOR PROCESSING TRANSACTION
NEC CORP
Inventor(s):HATANO YOJI
Application No. 2000276347 JP2000276347 JP, Filed 20000912,A1 Published 20020329Published 20020329

Abstract:
PROBLEM TO BE SOLVED: To provide a method and a system for processing transaction, by which operation management or transaction processing program can be easily customized by user and a client application can be easily prepared.
SOLUTION: The client/server type transaction processing system capable of online transaction processing, has a storage device for storing plural work components as a program for executing transaction processing, a work flow method as a program for calling and executing the work components and a work flow class as a program for storing the work flow method required for every work processing in a server node and a processor for calling the work component through the work flow method for the unit of work flow class and executing transaction processing according to this work component.
Int'l Class: G06F00946; G06F00944 G06F01200 G06F01516

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号
特開2002－91780
（P2002－91780A）
(43)公開日　平成14年３月29日(2002.3.29)

(51)Int.Cl.⁷		識別記号	FI		テーマコード(参考)
Ｇ０６Ｆ	9/46	３４０	Ｇ０６Ｆ 9/46	３４０Ｂ	５Ｂ０４５
	9/44		9/44	５３０Ｐ	５Ｂ０７６
		５３０	12/00	５１８Ａ	５Ｂ０８２
	12/00	５１８	15/16	６２０Ｆ	５Ｂ０９８
	15/16	６２０		６２０Ｔ	

審査請求　有　請求項の数６　ＯＬ　（全 13 頁）　最終頁に続く

(21)出願番号　特願2000－276347(P2000－276347)

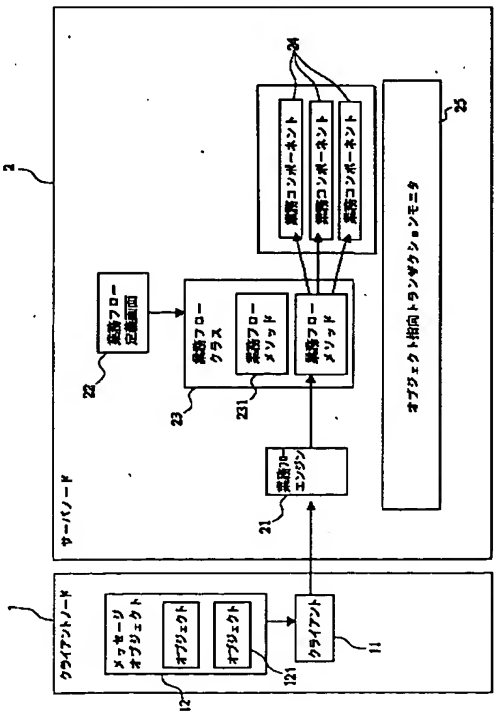
(22)出願日　平成12年９月12日(2000.9.12)

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　波多野　陽治
東京都港区芝五丁目７番１号　日本電気株式会社内
(74)代理人　100088328
弁理士　金田　暢之　（外２名）

Ｆターム(参考)　5B045 BB11 BB28 BB47 GG09
5B076 DB04 DC09 DD10
5B082 AA01 GB00
5B098 AA10 GA01 GC01 GC16 GD02
GD14

(54)【発明の名称】　トランザクション処理システム及び方法

(57)【要約】
【課題】　ユーザによる運用管理やトランザクション処理プログラムのカスタマイズが容易であり、クライアントアプリケーションを簡易に作成できるトランザクション処理方法及びシステムを提供する。
【解決手段】　オンラインでトランザクション処理が可能なクライアント／サーバ型のトランザクション処理システムであって、サーバノードに、トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な業務フローメソッドを格納するためのプログラムである業務フロークラスを格納する記憶装置と、業務フロークラス単位で、業務フローメソッドにより業務コンポーネントを呼び出し、該業務コンポーネントにしたがってトランザクション処理を実行する処理装置とを有する構成とする。



【特許請求の範囲】
【請求項１】　クライアントとして動作するクライアントノードと、
所定のトランザクション処理を実行するサーバノードとを有し、オンラインでトランザクション処理が可能なクライアント／サーバ型のトランザクション処理システムであって、
前記サーバノードは、
前記トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、前記業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な前記業務フローメソッドを格納するためのプログラムである業務フロークラスをそれぞれ格納する記憶装置と、
前記業務フロークラス単位で、前記業務フローメソッドにより前記業務コンポーネントを呼び出し、該業務コンポーネントにしたがって前記トランザクション処理を実行する処理装置と、を有するトランザクション処理システム。
【請求項２】　前記サーバノードは、
前記業務フロークラス単位で、前記業務コンポーネントの呼出し順序を定義するために書き換え可能な業務フロー定義画面を表示するための出力装置を有する請求項１記載のトランザクション処理システム。
【請求項３】　前記クライアントノードは、
前記サーバノードに業務処理を依頼するためのアプリケーションであるクライアントと、
前記クライアントノードから前記サーバノードに業務処理を依頼する際に前記業務コンポーネントに渡すオブジェクトを格納するためのプログラムであるメッセージオブジェクトと、をそれぞれ格納する記憶装置を有し、
前記メッセージオブジェクトに、前記業務コンポーネントのメソッドの各引数に対して付与された所定のキーと該キーに対応するオブジェクトとが表形式で格納される請求項１または２記載のトランザクション処理システム。
【請求項４】　クライアントとして動作するクライアントノードと、
所定のトランザクション処理を実行するサーバノードと、を有するクライアント／サーバシステムに実行させるためのトランザクション処理方法であって、
前記サーバノードに、前記トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、前記業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な前記業務フローメソッドを格納するためのプログラムである業務フロークラスを予め格納しておき、
前記業務フロークラス単位で、前記業務フローメソッドにより前記業務コンポーネントを呼び出し、該業務コンポーネントにしたがって前記トランザクション処理を実行するトランザクション処理方法。
【請求項５】　前記業務フロークラス単位で、前記業務コンポーネントの呼出し順序を定義するために書き換え可能な業務フロー定義画面を表示する請求項１記載のトランザクション処理方法。
【請求項６】　前記クライアントノードに、前記業務コンポーネントのメソッドの各引数に対して付与された所定のキーと該キーに対応するオブジェクトとを表形式から成るメッセージオブジェクトとして予め格納しておき、
前記クライアントノードから前記サーバノードに業務処理を依頼する際に、前記メッセージオブジェクトを参照し、前記業務コンポーネントに前記トランザクション処理で用いるオブジェクトを渡す請求項１または２記載のトランザクション処理方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明はオンラインでトランザクション処理が可能なクライアント／サーバ型のトランザクション処理システムに関する。
【０００２】
【従来の技術】従来、オープン系のトランザクション処理システムは、トランザクションモニタ上で構築され、トランザクション処理プログラムはトランザクションモニタ上で動作するライブラリにより作成されてきた。
【０００３】最近の動向として、トランザクションモニタがオブジェクト指向のコンポーネントの動作環境を提供するようになり、トランザクション処理プログラムは、ＥＪＢ（Enterprise Java Beans）、ＣＯＭ（Component Object Model）、ＣＯＲＢＡ（Common Object Request Broker Architecture）等によるコンポーネントとして作成される。こうしたトランザクションモニタを以下ではオブジェクト指向トランザクションモニタと呼ぶ。
【０００４】上述したようにコンポーネントを用いてトランザクション処理プログラムを作成すると、プログラムが部品化されるため再利用性を高めることができる。例えば、オブジェクト指向トランザクションモニタ上で作成したＥＪＢコンポーネントは、このオブジェクト指向トランザクションモニタ特有のＡＰＩ（Application Program Interface）等を使用していない限り、他のオブジェクト指向トランザクションモニタでも動作させることができる。例えば、ＢＥＡ　Ｓｙｓｔｅｍｓ社のＷｅｂＬｏｇｉｃ　Ｓｅｒｖｅｒの場合、コンポーネントにＪ２ＥＥ（Java 2 Enterprise Edition：サンマイクロシステムズ社）仕様で決められたＥＪＢをサポートしているため、ＥＪＢでトランザクション処理プログラムを作成することが可能である。これにより、ＷｅｂＳｈｏｐ等のＷｅｂブラウザから投入されたトランザクションをＷｅｂサーバ上のサーブレットに渡し、サーブ

レットでＷｅｂＬｏｇｉｃ　Ｓｅｒｖｅｒ上のＥＪＢで作成されたコンポーネント群を呼び出すことによりトランザクション処理を実行するシステム等を構築することができる。

【０００５】このようなトランザクション処理プログラムは一般に複数のコンポーネントの呼出し処理により構成される。通常、コンポーネントの粒度は任意であり、コンポーネントは必ずしもユーザにとって扱いやすい大きさであるとは限らない。また、オブジェクト指向トランザクションモニタでネイティブに提供する運用管理ツール等もコンポーネント単位で提供されるため、ユーザにとって運用管理が困難な場合がある。例えば、トランザクション処理プログラムの稼働状況を表示させる場合、従来のオブジェクト指向トランザクションモニタの運用管理ツールでは、コンポーネントのメソッドが１秒当たり何件実行されたか等の情報を表示するだけであるため、ユーザにとってわかりやすいものではない。

【０００６】このような問題を解決するため、例えば、特開２０００－１３２３９４号公報では、オブジェクトを、処理フローを制御する制御オブジェクト、及びデータアクセスのインターフェースを実装するオブジェクト管理等に分類してオブジェクト指向業務システムを構築していく指針が提示されている。この場合、設計段階から所定の指針にしたがって統一した設計を行えば、オブジェクトの粒度、境界が不均一になる等の問題が発生せず、制御オブジェクトをユーザの業務に対応したものにすることができる。

【０００７】

【発明が解決しようとする課題】しかしながら、上記特開２０００－１３２３９４号公報に記載された技術では、既に稼働している業務システムに対して適用することができないという問題がある。特に、近年は外部コンポーネントの流通が見込まれるため、既存のコンポーネントをまとめる技術も必要とされる。

【０００８】加えて、制御オブジェクトやオブジェクト管理に分類してトランザクション処理プログラムを作成する場合でも、異常時の終了処理やコンポーネント呼出し前処理／後処理等のように、ユーザが独自の処理をトランザクション処理のフローに容易に追加できるものではなかった。

【０００９】また、上述したように、トランザクション処理システムでは、コンポーネントの粒度が任意であり、必ずしも運用するユーザにとってわかりやすい大きさではないため、運用管理が難しいという問題がある。

【００１０】さらに、トランザクション処理が複数のコンポーネント呼び出し処理で構成されている場合、コンポーネントの呼び出しツールやコンポーネントの呼び出し結果を他のコンポーネントに渡す際のデータ変換ロジック等をクライアント側で実装しているため、クライアント側のアプリケーションの作成負荷が大きくなるという問題があった。

【００１１】本発明は上記したような従来の技術が有する問題点を解決するためになされたものであり、ユーザによる運用管理やトランザクション処理プログラムのカスタマイズが容易であり、クライアントアプリケーションを簡易に作成できるトランザクション処理方法及びシステムを提供することを目的とする。

【００１２】

【課題を解決するための手段】上記目的を達成するため本発明のトランザクション処理システムは、クライアントとして動作するクライアントノードと、所定のトランザクション処理を実行するサーバノードとを有し、オンラインでトランザクション処理が可能なクライアント／サーバ型のトランザクション処理システムであって、前記サーバノードは、前記トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、前記業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な前記業務フローメソッドを格納するためのプログラムである業務フロークラスをそれぞれ格納する記憶装置と、前記業務フロークラス単位で、前記業務フローメソッドにより前記業務コンポーネントを呼び出し、該業務コンポーネントにしたがって前記トランザクション処理を実行する処理装置と、を有する構成である。

【００１３】このとき、前記サーバノードは、前記業務フロークラス単位で、前記業務コンポーネントの呼出し順序を定義するために書き換え可能な業務フロー定義画面を表示するための出力装置を有していてもよく、前記クライアントノードは、前記サーバノードに業務処理を依頼するためのアプリケーションであるクライアントと、前記クライアントノードから前記サーバノードに業務処理を依頼する際に前記業務コンポーネントに渡すオブジェクトを格納するためのプログラムであるメッセージオブジェクトと、をそれぞれ格納する記憶装置を有し、前記メッセージオブジェクトに、前記業務コンポーネントのメソッドの各引数に対して付与された所定のキーと該キーに対応するオブジェクトとが表形式で格納されていてもよい。

【００１４】一方、本発明のトランザクション処理方法は、クライアントとして動作するクライアントノードと、所定のトランザクション処理を実行するサーバノードと、を有するクライアント／サーバシステムに実行させるためのトランザクション処理方法であって、前記サーバノードに、前記トランザクション処理を実行するためのプログラムである複数の業務コンポーネント、前記業務コンポーネントを呼び出して実行するためのプログラムである業務フローメソッド、及び業務処理毎に必要な前記業務フローメソッドを格納するためのプログラムである業務フロークラスを予め格納しておき、前記業務フロークラス単位で、前記業務フローメソッドにより前

記業務コンポーネントを呼び出し、該業務コンポーネントにしたがって前記トランザクション処理を実行する方法である。

【００１５】このとき、前記業務フロークラス単位で、前記業務コンポーネントの呼出し順序を定義するために書き換え可能な業務フロー定義画面を表示してもよく、前記クライアントノードに、前記業務コンポーネントのメソッドの各引数に対して付与された所定のキーと該キーに対応するオブジェクトとを表形式から成るメッセージオブジェクトとして予め格納しておき、前記クライアントノードから前記サーバノードに業務処理を依頼する際に、前記メッセージオブジェクトを参照し、前記業務コンポーネントに前記トランザクション処理で用いるオブジェクトを渡してもよい。

【００１６】上記のようなトランザクション処理システム及び方法では、業務フロークラス単位で、業務フローメソッドにより業務コンポーネントを呼び出し、該業務コンポーネントにしたがってトランザクション処理を実行することで、稼動状況やその他物件管理などの各種運用単位を、ユーザにとって分かりやすい業務単位で行うことができる。

【００１７】

【発明の実施の形態】次に本発明について図面を参照して説明する。

【００１８】図１は本発明のトランザクション処理システムの一構成例を示すブロック図である。

【００１９】図１において、本発明のトランザクション処理システムは、クライアントとして動作するクライアントノード１と、コンポーネントにしたがってトランザクション処理を実行するサーバノード２とを有する構成である。

【００２０】クライアントノード１は、例えば、パーソナルコンピュータ等の情報処理装置を用いて構成され、サーバノード２に業務処理を依頼するためのアプリケーションであるクライアント１１と、クライアント１１からサーバノード２に業務処理を依頼する際に、トランザクション処理を実行する業務コンポーネントに渡されるオブジェクト（コンポーネント）１２１を格納するためのプログラムであるメッセージオブジェクト１２とを備えている。

【００２１】メッセージオブジェクト１２は、ハッシュテーブル構造を備えており、例えば、プログラム言語がＪａｖａの場合、HashTable Classをオブジェクト化したものが使用される。メッセージオブジェクト１２には、所定のキーとそれに対応する値とが格納され、値として任意のオブジェクトを指定することが可能である。ここでは、オブジェクト１２１が指定される。

【００２２】一方、サーバノード２は、例えば、ワークステーション等の情報処理装置によって構成され、トランザクション処理を実行するためのプログラムである複数の業務コンポーネント２４と、ユーザによって定義された業務フローにしたがって業務コンポーネント２４を呼び出して実行するためのプログラムである業務フローメソッド２３１と、業務処理毎に、必要な複数の業務フローメソッド２３１を格納するためのプログラムである業務フロークラス２３と、クライアント１１からの要求を受け付け、クライアント１１により指定された業務フロークラス２３による処理を実行させるためのプログラムである業務フローエンジン２１とを備えている。なお、業務コンポーネント２４は、実際にはサーバノード２が有するオブジェクト指向トランザクションモニタ２５上で実行される。また、図１では業務フロークラス２３が１つだけ記載されているが、業務の種類に応じて複数の業務フロークラス２３を備えていてもよい。

【００２３】業務コンポーネント２４は、ＥＪＢ、ＣＯＲＢＡ等によりユーザが作成するコンポーネントであり、これらの業務コンポーネントによりトランザクション処理が実行される。すなわち、トランザクション処理はこれら複数の業務コンポーネントの呼出し処理で構成される。

【００２４】業務フローエンジンは２１は、サーバノード２にコンポーネントとして実装され、業務フローエンジン２１のメソッドは、周知のコンポーネント呼出しツールによってクライアント１１から呼び出される。

【００２５】業務フローは、業務コンポーネント２４の呼出し順序を定義したものであり、業務フロー定義画面２２を用いてユーザにより登録される。また、この業務フロー定義画面２２を用いて登録された業務フローからサーバノード２により業務フロークラス２３が構成される。

【００２６】クライアントノード１あるいはサーバノード２となる情報処理装置は、例えば、図２に示すように、プログラムにしたがって所定の処理を実行する処理装置１１０と、処理装置１１０に対してコマンドや情報等を入力するための入力装置１２０と、処理装置１１０の処理結果をモニタするための出力装置１３０とを有する構成である。

【００２７】処理装置１１０は、ＣＰＵ１１１と、ＣＰＵ１１１の処理に必要な情報を一時的に記憶する主記憶装置１１２と、ＣＰＵ１１１に本発明の業務システムの処理を実行させるためのプログラムが記録された記録媒体１１３と、クライアントノード１としての処理やサーバノード２としての処理に必要なプログラムやデータが格納されるデータ蓄積装置１１４と、主記憶装置１１２、記録媒体１１３、及びデータ蓄積装置１１４とのデータ転送を制御するメモリ制御インタフェース部１１５と、入力装置１２０及び出力装置１３０とのインタフェース装置であるＩ／Ｏインタフェース部１１６と、外部ノードとのデータ通信を制御するインタフェースである通信制御装置１１７とを備え、それらがバス１１８を介

して接続された構成である。

【００２８】処理装置１１０は、記録媒体１１３に記録されたプログラムにしたがって以下に記載する本発明のトランザクション処理システムの処理を実行する。なお、記録媒体１１３は、磁気ディスク、半導体メモリ、光ディスクあるいはその他の記録媒体であってもよい。

【００２９】次に、図１に示したトランザクション処理システムの動作について詳細に説明する。

【００３０】本発明のトランザクション処理システムでは、トランザクション処理を実行するコンポーネントを「業務」単位でまとめる。これは以下の考えにしたがって行う。

【００３１】一般に、「業務」は複数のトランザクション処理から構成される。例えば、普通預金業務は、入金トランザクション処理、及び出金トランザクション処理等を含んでいる。これらのトランザクション処理では、通常、コンポーネントの呼び出し処理が複数回実行される。

【００３２】本発明のトランザクション処理システムでは、このような「業務」に必要な各トランザクション処理に対応する業務フローメソッド２３１を定義し、複数の業務フローメソッド２３１を備えた「業務」に対応するクラスを定義し、各クラス毎に業務フロークラス２３を作成する。

【００３３】本発明のトランザクション処理システムの処理内容は、業務フローの定義処理と実行処理とに分離される。まず業務フローの定義処理について図３及び図４を用いて説明する。

【００３４】図３は図１に示したトランザクション処理システムの処理手順を示す図であり、業務フローの定義処理の手順を示すフローチャートである。また、図４は図１に示した業務フロー定義画面の一例を示すイメージ図である。

【００３５】図３に示すように、業務フローの定義処理では、ユーザは、サーバノード２により業務フロー定義画面２２を起動し（ステップＡ１）、業務フロー定義画面２２の指示にしたがって、まず、業務フローで呼び出す業務コンポーネント２４、及びそのメソッドをそれぞれ選択して登録する（ステップＡ２）。ここで、業務コンポーネント２４のメソッドにはそれぞれ複数の引数が設けられているため、各々の引数に所定のキーを指定する（ステップＡ３）。これらのキーはオブジェクト１２１に対応するものであり、クライアント１１から送信されるメッセージオブジェクト１２からオブジェクト１２１を取り出す際に必要となる。

【００３６】続いて、業務コンポーネント２４の呼び出し処理の前後で何らかの処理が必要となる場合は、呼び出し前処理／後処理（データ変換規則等）を定義する（ステップＡ４）。そして、業務フローに挿入すべき業務コンポーネント２４がまだあるか否かを判定し、業務コンポーネント２４がある場合はステップＡ２の処理に戻ってステップＡ４までの処理を繰り返し、業務フローを作成する（ステップＡ５）。

【００３７】業務フローの作成が完了したら、業務コンポーネント２４を呼び出す業務フロークラス２３を作成する（ステップＡ６）。例えば、プログラム言語がＪａｖａの場合、これはクラスファイルとなる。作成した業務フロークラス２３はサーバノード２に格納される。

【００３８】次に、業務フローの実行処理について図５を参照して説明する。

【００３９】図５は図１に示したトランザクション処理システムの処理手順を示す図であり、業務フローの実行処理の手順を示すシーケンス図である。

【００４０】図５において、ユーザは、予めクライアント１１を用いて業務コンポーネント２４に渡すオブジェクト１２１を作成し、それらをメッセージオブジェクト１２にハッシュテーブル形式でクライアントノード１に格納しておく。

【００４１】このような状態で、クライアントノード１からサーバノード２にオンライントランザクション処理を依頼する場合、ユーザは、まずクライアント１１を用いてサーバノード２にアクセスし、業務フローエンジン２１のメソッドを呼び出す。この時、クライアントノード１は、呼び出す業務フロークラス２３の名称、及びメッセージオブジェクト１２を業務フローエンジン２１に引数として渡す。

【００４２】サーバノード２は、業務フローエンジン２１によって指定された業務フロークラス２３をインスタンス化し、対応する業務フローメソッド２３１を呼び出す。そして、業務フローメソッド２３１により、業務フロー定義画面２２で登録された業務フローにしたがって指定された業務コンポーネント２４を順次呼び出し、業務コンポーネント２４によるトランザクション処理の結果を業務フローエンジン２１を介してクライアント１１へそれぞれ返却する。

【００４３】なお、業務コンポーネント２４を呼び出す際に、業務フロー定義画面２２で呼出し前処理が定義されている場合は、指定された前処理を行った後に業務コンポーネント２４を呼び出す。また、呼出し後処理が定義されている場合は、業務コンポーネント２４を実行した後に指定された後処理を行う。クライアントノード１は、クライアント１１により業務コンポーネント２４を呼び出す時に、業務フローの定義処理の際に設定されたキーにしたがって、メッセージオブジェクト１２に格納されたオブジェクト１２１を業務コンポーネント２４に渡していく。

【００４４】次に、メッセージオブジェクト１２に格納されたオブジェクト１２１を業務コンポーネント２４に渡す手順ついて図６を用いて説明する。

【００４５】図６は図１に示したクライアントノードの

メッセージオブジェクトからサーバノードの業務コンポーネントにオブジェクトを渡すときの手順を示す模式図である。

【００４６】図６において、業務フロー定義画面２２を用いて業務フローで呼び出す業務コンポーネント２４のメソッドを定義するとき、業務コンポーネント２４のメソッドの各引数に対応するキーが定義される。図６では、例えば、コンポーネントＡのメソッドＡの引数arg1に対して、CMPA#MethodA#arg1がキーとして定義されている。

【００４７】クライアントノード１は、クライアント１１によりメッセージオブジェクト１２を作成する時、業務コンポーネント２４に渡すオブジェクト１２１を、上記定義されたキーに基づいてメッセージオブジェクト１２に格納する。図６では、例えば、CMPA#MethodA#arg1をキーにしてObj-1がメッセージオブジェクト１２に格納される。

【００４８】サーバノード２は、業務フロークラス２３により業務コンポーネント２４を呼び出すとき、メソッドの引数に対して定義されたキーに対応するオブジェクト１２１をメッセージオブジェクト１２から取り出し、引数にセットして業務コンポーネント２４のメソッドを呼び出す。図６の場合、例えば、メッセージオブジェクト１２からObj-1、Obj-2を取り出し、Method Aのパラメータにセットして Component AのMethod Aを呼び出している。

【００４９】このような方法でメッセージオブジェクト１２に含まれる各オブジェクト１２１を業務コンポーネント２４に渡すことで、各オブジェクト１２１をそれぞれ適切な業務コンポーネント２４に渡すことができる。

【００５０】次に本発明の形態の効果について説明する。

【００５１】上述したように、業務コンポーネントの粒度は任意であるため、業務コンポーネントによる処理単位はユーザにとって分かりにくい場合がある。本発明のように業務フロークラス２３を定義することで、トランザクション処理がユーザにとって認識しやすいものとなり、運用管理が容易になる。

【００５２】例えば、業務処理の稼働状況を表示する場合を例にして以下に説明する。

【００５３】通常、オブジェクト指向トランザクションモニタによる稼動状況の認識単位はコンポーネントであり、オブジェクト指向トランザクションモニタに付属した管理ツール等を用いて確認できる稼働状況は、図７（ａ）に示すように、コンポーネントのメソッド呼び出し回数等であり、ユーザにとってわかりにくいものである。

【００５４】本発明の場合、例えば、業務フロークラス２３の呼び出し回数をカウントして表示すれば、ユーザは、図７（ｂ）に示すように、稼働状況を業務フロークラス単位、すなわち業務単位で確認することが可能になる。また、その他物件管理などの各種運用単位を業務毎に行うことができる。

【００５５】また、業務や業務フローが単に論理的な単位でなく、クラス、クラスのメソッドというユーザに認識しやすい物理的な実体にマッピングされているため、トランザクション処理プログラムにユーザ独自の処理プログラムを容易に追加することができる。例えば、コンポーネント呼び出し前に独自のロジックを組み込んだり、異常終了時の処理の追加やループの作成等のカスタマイズを容易に行うことができる。また、カスタマイズを業務フロー定義画面を用いて行えるため、カスタマイズがより容易になる。

【００５６】さらに、本発明では、オンラインによるトランザクション処理で多数のコンポーネント呼出し処理が発生する場合でも、クライアント１１は業務フローエンジン２１のメソッドを呼び出して１つのオブジェクト１２１を渡す処理だけで済むため、クライアントアプリケーションの簡易化が可能になる。

【００５７】次に、本発明の応用例について図面を参照して説明する。

【００５８】本発明は主としてコンポーネントベースでトランザクション処理を実行するトランザクション処理システムに関するものであるが、コンポーネントベースでないインターフェースを備えたトランザクション処理システムについても適用可能である。

【００５９】例えば、図８に示すように、ＥＡＩツール３１等を使用してＥＡＩツールに対応するラッパーコンポーネントを業務フローメソッドで呼び出すことにより、コンポーネントベースでないインターフェースを備えたトランザクション処理システムを業務フローの一部に組み込むことができる。また、ＥＡＩツール３１によりデータをＸＭＬ形式に変換することでロゼッタネット３４への適用も可能である。すなわち、コンポーネントベースではないトランザクション処理システムも業務単位にまとめることが可能になる。

【００６０】図８に示すトランザクション処理システムは、図１に示したトランザクション処理システムに、異種システムとの接続処理、あるいはデータ形式変換処理などを行って異種システムとの連携を可能にするＥＡＩ（Enterprise ApplicationIntegration）ツール３１と、ＥＡＩツール３１をコンポーネントとして取り扱うためのコンポーネントインターフェースを提供するラッパーコンポーネント３２と、クライアントノードにインターネットを介して接続されたＷｅｂブラウザ３３とをさらに有する構成である。その他の構成は図１に示したトランザクション処理システムと同様であるため、その説明は省略する。

【００６１】ＥＡＩツール３１は、メッセージキューイングシステム、ＥＲＰ（Enterprise Resource Plannin

ｇ）等の異なるインターフェースを有するシステムを簡単に接続するためのツールであり、代表的なものとして、Ｍｅｒｃａｔｏｒ(Ｍｅｒｃａｔｏｒ社)、ｅ＊Ｇａｔｅ(ＳＴＣ社)等がある。

【００６２】また、ラッパーコンポーネント３２により異種システムもコンポーネントとして業務フローメソッド２３１で呼び出すことが可能であり、ユーザは異種システムもトランザクション処理システムに組み込むことが可能になる。

【００６３】具体的には、インターネットを介してＷｅｂブラウザ３３から電子商取引関連の注文がクライアントノードに発注され、クライアントからの依頼によりサーバノードの業務フローメソッドを実行させる。このとき、業務フローメソッドは、自サーバノードに格納された業務コンポーネントを呼び出した後、ラッパーコンポーネント３２を呼び出して、ＥＡＩツール３１によりデータをＸＭＬ形式に変換し、Ｅ－Ｍａｒｋｅｔ　Ｐｌａｃｅ（ロゼッタネット）に流すようなトランザクション処理が考えられる。このような一連のトランザクション処理を１つの業務として認識することができるため、稼働状況等を業務単位で表示させることができる。

【００６４】

【発明の効果】本発明は以上説明したように構成されているので、以下に記載する効果を奏する。

【００６５】業務フロークラス単位で、業務フローメソッドにより業務コンポーネントを呼び出し、該業務コンポーネントにしたがってトランザクション処理を実行することで、稼動状況やその他物件管理などの各種運用単位をユーザにとって分かりやすい業務単位で行うことができる。また、業務や業務フローが単に論理的な単位でなく、クラス、クラスのメソッドというユーザに認識しやすい物理的な実体にマッピングされるため、トランザクション処理プログラムにユーザ独自の処理プログラムを容易に追加することができる。

【００６６】また、ユーザによるトランザクション処理プログラムのカスタマイズを業務フロー定義画面を用いて行えば、カスタマイズがより容易になる。

【００６７】さらに、オンラインによるトランザクション処理で多数のコンポーネント呼出し処理が発生する場合でも、クライアントノードは、サーバノードに格納された業務コンポーネントに、対応する１つのオブジェクトを渡す処理だけで済むため、クライアントアプリケーションの簡易化が可能になる。

【図面の簡単な説明】

【図１】本発明のトランザクション処理システムの一構成例を示すブロック図である。

【図２】図１に示したクライアントノード及びサーバノードの一構成例を示すブロック図である。

【図３】図１に示したトランザクション処理システムの処理手順を示す図であり、業務フローの定義処理の手順を示すフローチャートである。

【図４】図１に示した業務フロー定義画面の一例を示すイメージ図である。

【図５】図１に示したトランザクション処理システムの処理手順を示す図であり、業務フローの実行処理の手順を示すシーケンス図である。

【図６】図１に示したクライアントノードのメッセージオブジェクトからサーバノードの業務コンポーネントにオブジェクトを渡すときの手順を示す模式図である。

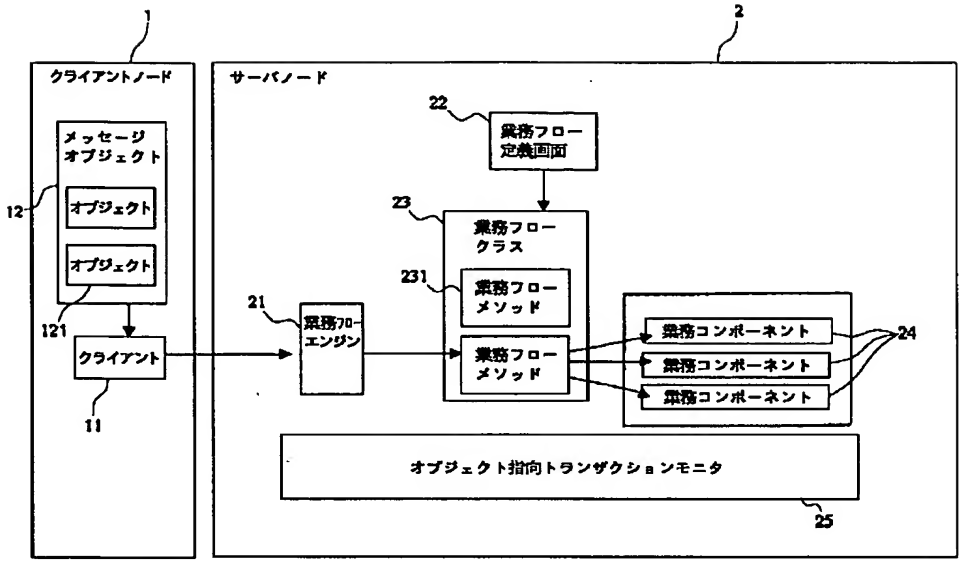
【図７】オブジェクト指向トランザクションモニタに付属した管理ツール等を用いて確認できる稼働状況を示す図であり、同図（ａ）は従来の表示内容を示す表であり、同図（ｂ）は本発明の表示内容を示す表である。

【図８】本発明のトランザクション処理システムの他の構成例を示すブロック図である。

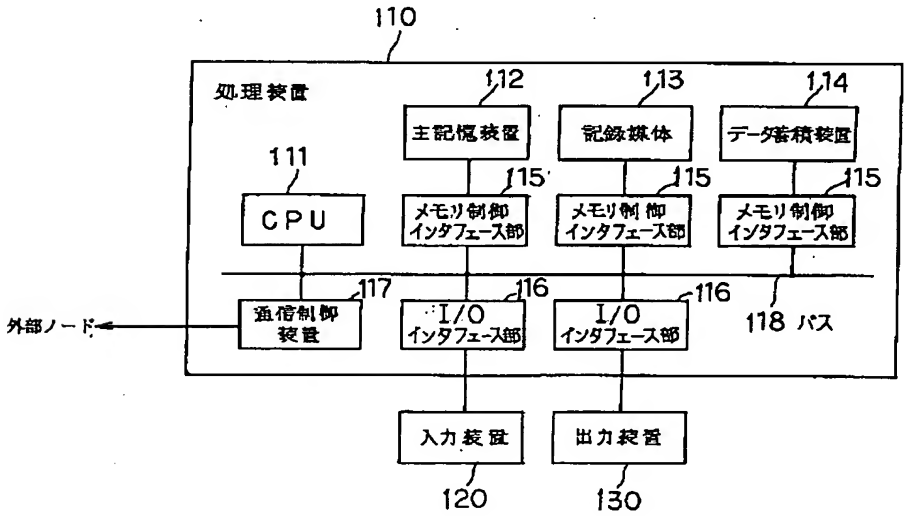
【符号の説明】

１　クライアントノード
２　サーバノード
１１　クライアント
１２　メッセージオブジェクト
２１　業務フローエンジン
２２　業務フロー定義画面
２３　業務フロークラス
２４　業務コンポーネント
２５　オブジェクト指向トランザクションモニタ
３１　ＥＡＩツール
３２　ラッパーコンポーネント
３３　Webブラウザ
１１０　処理装置
１１１　ＣＰＵ
１１２　主記憶装置
１１３　記録媒体
１１４　データ蓄積装置
１１５　メモリ制御インタフェース部
１１６　Ｉ／Ｏインタフェース部
１１７　通信制御装置
１１８　バス
１２０　入力装置
１２１　オブジェクト
１３０　出力装置
２３１　業務フローメソッド

【図１】



【図２】



【図７】

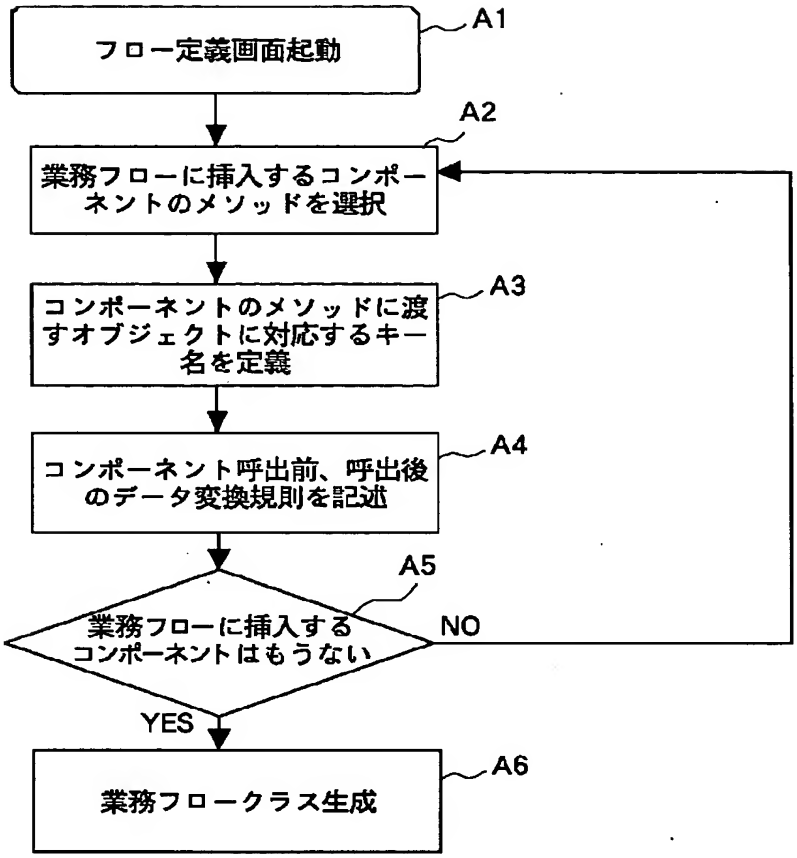
(a)

コンポーネント、メソッド	回数
Comp1, Method1	10回／秒
Comp1, Method2	10回／秒

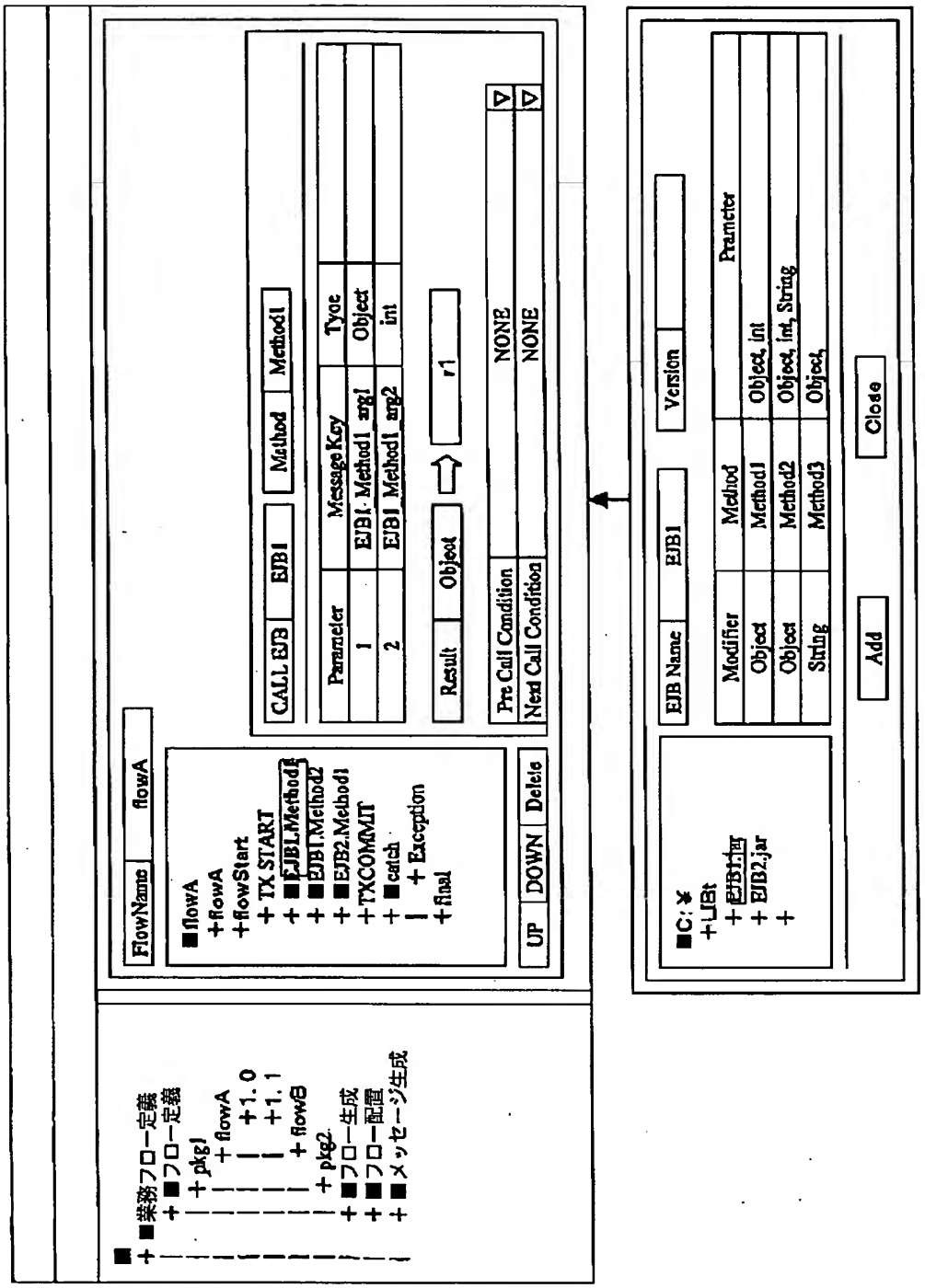
(b)

コンポーネント、メソッド	回数
普通預金業務	20件／秒
定期預金業務	10回／秒

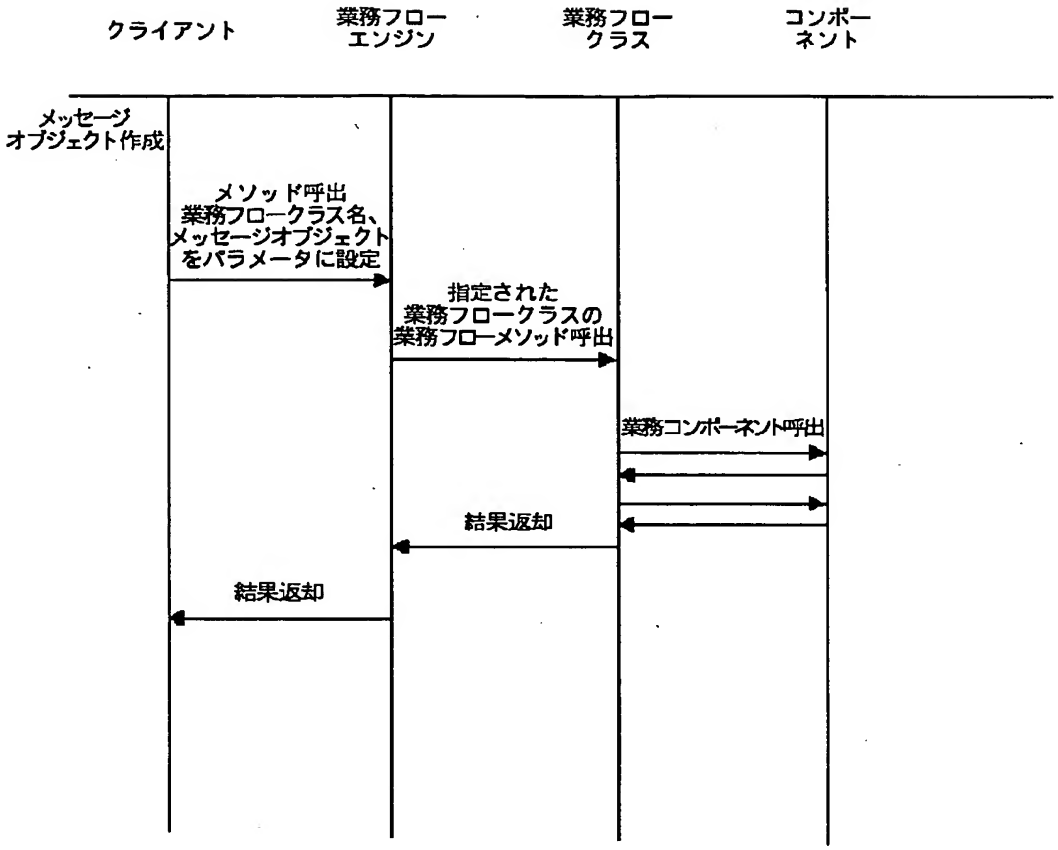
【図３】


フロー定義画面起動
A1
業務フローに挿入するコンポーネントのメソッドを選択
A2
コンポーネントのメソッドに渡すオブジェクトに対応するキー名を定義
A3
コンポーネント呼出前、呼出後のデータ変換規則を記述
A4
業務フローに挿入するコンポーネントはもうない
A5
NO
YES
業務フロークラス生成
A6

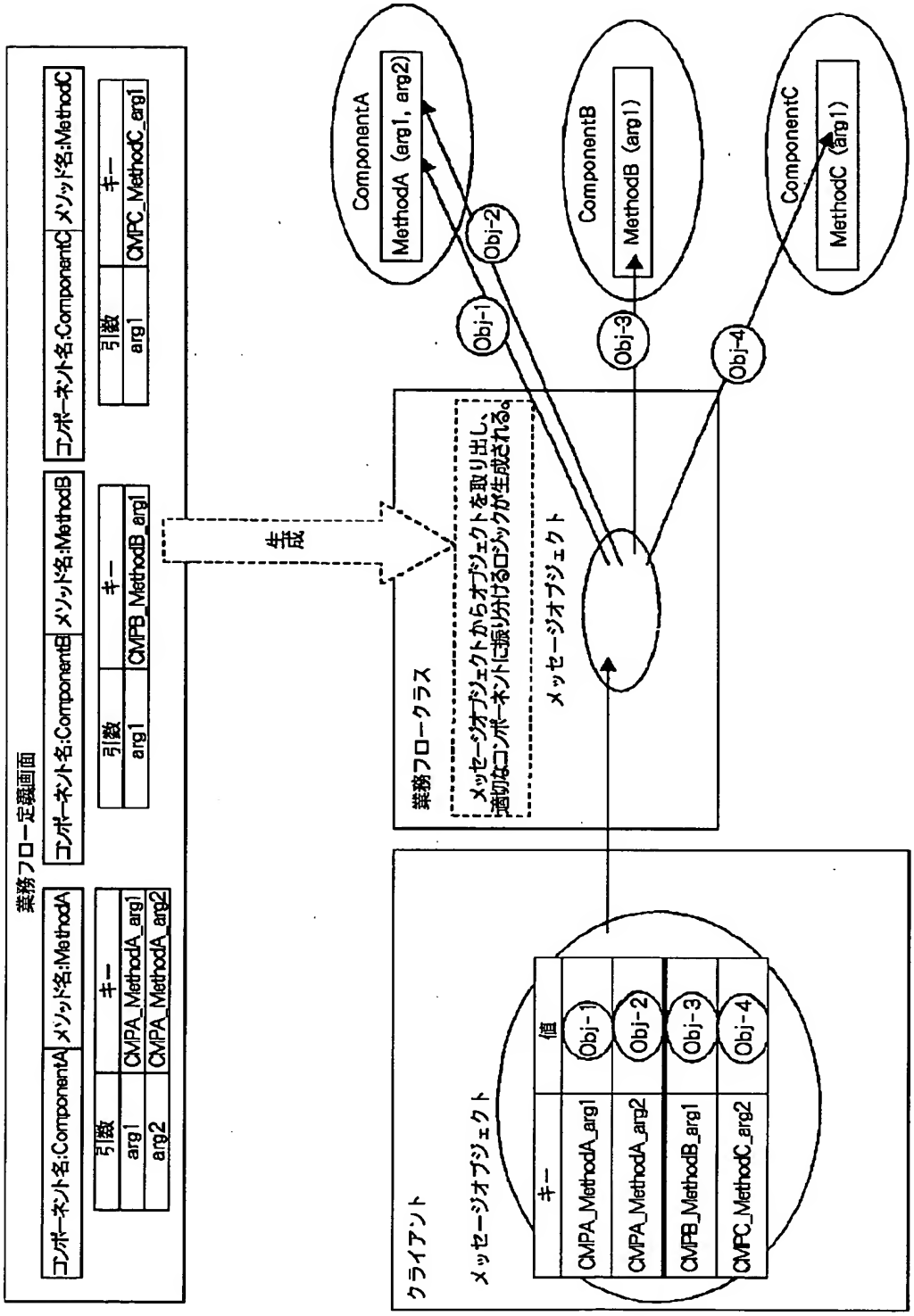
【図４】


■
+■業務フロー定義
+■フロー定義
+pkg1
+flowA
+1.0
+1.1
+flowB
+pkg2
+■フロー生成
+■フロー配置
+■メッセージ生成
FlowName
flowA
■flowA
+flowA
+flowStart
+TX START
+■EJB1.Method1
+■EJB1.Method2
+■EJB2.Method1
+TXCOMMIT
+■catch
+Exception
+final
UP
DOWN
Delete
CALL EJB
EJB1
Method
Method1
Parameter
Message Key
Type
1
EJB1.Method1_arg1
Object
2
EJB1.Method1_arg2
int
Result
Object
r1
Pre Call Condition
NONE
Next Call Condition
NONE
■C:¥
+LIBR
+EJB1.jar
+EJB2.jar
+
EJB Name
EJB1
Version
Modifier
Method
Parameter
Object
Method1
Object, int
Object
Method2
Object, int, String
String
Method3
Object,
Add
Close

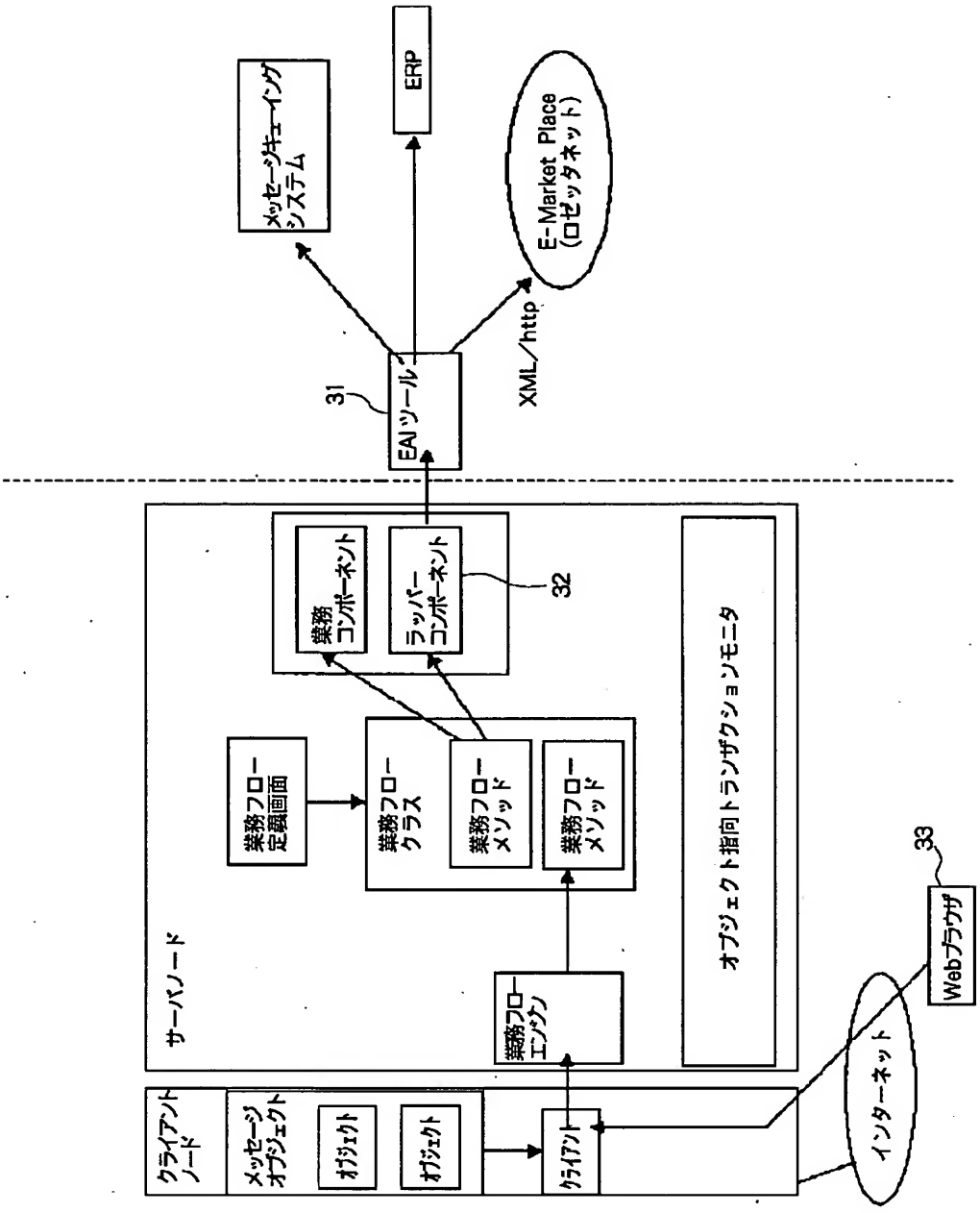
【図５】


クライアント
業務フロー
エンジン
業務フロー
クラス
コンポー
ネント
メッセージ
オブジェクト作成
メソッド呼出
業務フロークラス名、
メッセージオブジェクト
をパラメータに設定
指定された
業務フロークラスの
業務フローメソッド呼出
業務コンポーネント呼出
結果返却
結果返却

【図６】


業務フロー定義画面
コンポーネント名:ComponentA
メソッド名:MethodA
引数
arg1
arg2
キー
CMPA_MethodA_arg1
CMPA_MethodA_arg2
コンポーネント名:ComponentB
メソッド名:MethodB
引数
arg1
キー
CMPB_MethodB_arg1
コンポーネント名:ComponentC
メソッド名:MethodC
引数
arg1
キー
CMPC_MethodC_arg1
生成
業務フロークラス
メッセージオブジェクトからオブジェクトを取り出し、適切なコンポーネントに振り分けるロジックが生成される。
メッセージオブジェクト
ComponentA
MethodA (arg1, arg2)
ComponentB
MethodB (arg1)
ComponentC
MethodC (arg1)
Obj-1
Obj-2
Obj-3
Obj-4
クライアント
メッセージオブジェクト
キー
値
CMPA_MethodA_arg1
Obj-1
CMPA_MethodA_arg2
Obj-2
CMPB_MethodB_arg1
Obj-3
CMPC_MethodC_arg2
Obj-4

【図８】



フロントページの続き

(51)Int.Cl.[7]	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　15/16	６２０	Ｇ０６Ｆ　9/06	６２０Ａ